



Modélisation des mouvements explicites dans les ACG avec le produit dépendant

Florent Pompigne

► To cite this version:

Florent Pompigne. Modélisation des mouvements explicites dans les ACG avec le produit dépendant. Conférence sur le Traitement Automatique des Langues Naturelles - TALN 2009 - RECITAL, LIPN, Jun 2009, Senlis, France. inria-00441912

HAL Id: inria-00441912

<https://inria.hal.science/inria-00441912>

Submitted on 17 Dec 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Modélisation des mouvements explicites dans les ACG avec le produit dépendant

Florent Pompigne¹

(1) ENS Cachan / INRIA Nancy - Grand-Est

florent.pompigne@loria.fr

Abstract. Abstract Categorical Grammars (ACG) is a grammatical framework based on linear lambda-calculus. As in Muskens' Lambda Grammars, an abstract term in this kind of categorical grammar can be realized in different directions, such as syntactic and semantic ones. This structure provides autonomy for these different processings. ACG's architecture is independent from the logic used and so the type system is easily extensible in order to deal better with some linguistic phenomena. We will first introduce ACGs and the dependent product construction. This paper will then be concerned with the issue of overt grammatical movements, in particular extraction constraints in relative propositions, and how several close frameworks deal with it. Last we will show how to capture this phenomenon in extended ACG.

Mots-clés : syntaxe, grammaires catégorielles abstraites, types dépendant, mouvements explicites, extraction.

Keywords: Syntax, abstract categorical grammars, dependant product, overt movements, extraction.

1 Introduction

La description de la syntaxe d'une langue naturelle et de la relation entre ce niveau syntaxique et le niveau sémantique est traditionnellement opérée en Traitement Automatique des Langues par un formalisme grammatical. Les Grammaires Catégorielles Abstraites (ACG), proposées par de Groote (de Groote, 2001), se distinguent dans le paysage des formalismes existants par des propriétés héritées des grammaires catégorielles (sensibilité aux ressources) et des propriétés propres à leur architecture : elles s'appuient sur un langage abstrait, dont les termes correspondent à des structures de dérivation, et un langage objet dont les termes correspondent aux structures dérivées.

Les mouvements grammaticaux explicites désignent les déplacements qui ont lieu dans des constructions comme les formes interrogatives ou les propositions relatives par rapport aux phrases correspondantes. L'extraction d'un composant de la phrase par ces constructions peut s'effectuer à distance arbitraire, et selon des contraintes précises de la langue qui sont bien étudiées et transcrites dans de nombreux formalismes grammaticaux. Le propos de cet article est de présenter une modélisation de ce phénomène pour les ACG.

Dans la partie suivante on introduira le formalisme des ACG ainsi que l'extension du système de type avec le produit dépendant. Dans la partie 3 sera présenté le problème des mouvements explicites, à travers les contraintes à l'extraction dans les constructions relative en français, et les réponses apportées dans plusieurs formalismes voisins, et la partie 4 détaillera la solution proposée pour les ACG.

2 Grammaires Catégorielles Abstraites

Avec une approche similaire aux Lambda Grammars (Muskens, 2007) ou aux Convergent Grammars (Pollard, 2007) , (Pollard, 2008), chaque ACG introduit une fonction d'un espace de termes dits abstraits vers un espace de termes objets. Ces termes objets peuvent représenter différentes structures de données utilisées en traitement automatique des langues, comme des arbres ou des chaînes de caractère, et on choisit donc de les construire sur le λ -calcul linéaire.

2.1 Architecture du formalisme

Intuitivement, une ACG consiste en deux signatures, appelées vocabulaire abstrait et vocabulaire objet, qui déclarent des types et des constantes typées dans le fragment implicatif de la logique linéaire, et d'un homomorphisme, appelé lexique, de la première vers la seconde. Une Grammaire Catégorielle Abstraite génère deux langages, un langage abstrait et un langage objet. Le langage abstrait est défini comme l'ensemble des λ -termes linéaires (chaque abstraction lie une unique variable) clos construits sur le vocabulaire abstrait qui sont du type spécifique S . Le langage objet est alors l'image du langage abstrait par le lexique.

Cette architecture permet à l'aide de deux ACG partageant un même langage abstrait (correspondant au niveau tectogrammatique) de réaliser ces termes abstraits vers un langage décrivant

leur sémantique et un langage décrivant leur forme de surface (correspondant au niveau phé-nogrammatique). Nous ne nous intéressons uniquement dans ce papier à la description de la syntaxe et donc à la relation entre tectogrammatique et phénogrammatique. Donnons un exemple d'une telle ACG :

N, NP, S : *type* (vocabulaire abstrait)

$homme_c, village_c, ami_c : \mathbf{N}$

$joyeux_c : \mathbf{N} \multimap \mathbf{N}$

$un_c, le_c : \mathbf{N} \multimap \mathbf{NP}$

$dormir_c : \mathbf{NP} \multimap \mathbf{S}$

$aimer_c, rencontrer_c : \mathbf{NP} \multimap \mathbf{NP} \multimap \mathbf{S}$

$Jean_c, Marie_c, Paul_c : \mathbf{NP}$

$de_c : \mathbf{NP} \multimap \mathbf{N} \multimap \mathbf{N}$

string : *type* (vocabulaire objet)

$homme, joyeux, un, \dots : \mathbf{string}$

N, NP, S \longrightarrow **string** (lexique)

$homme_c \longrightarrow homme$

$joyeux_c \longrightarrow \lambda x.x + joyeux$

$un_c \longrightarrow \lambda x.un + x$

$dormir_c \longrightarrow \lambda x.x + dort$

$aimer_c \longrightarrow \lambda x,y.y + aime + x$

$Jean_c \longrightarrow Jean$

$de_c \longrightarrow \lambda x,y.y + de + x$

(on note par "+" l'opération de concaténation des chaînes de caractère)

On peut vérifier que le terme suivant appartient au langage abstrait

$$\underbrace{\underbrace{rencontrer_c \left(\underbrace{un_c \left(\underbrace{de_c \left(Marie_c \text{ } ami_c \right)}_{\mathbf{N}} \right)}_{\mathbf{NP}} \right)}_{\mathbf{NP}} Jean_c}_{\mathbf{S}}$$

Son image par le lexique fait donc partie du langage objet

$rencontrer_c (un_c (de_c Marie_c \text{ } ami_c)) Jean_c$

$\longrightarrow (\lambda x,y.y + rencontre + x) ((\lambda x.un + x) ((\lambda x,y.y + de + x) Marie \text{ } ami)) Jean$

$\longrightarrow Jean \text{ } rencontre \text{ } un \text{ } ami \text{ } de \text{ } Marie$ (β -réduction)

A l'inverse, la possibilité de partir d'un terme de surface et de décider s'il appartient au langage d'une ACG est un problème ouvert équivalent à la décidabilité du fragment exponentiel multiplicatif de la logique linéaire (Salvati, 2005). Cependant, si on se restreint à la classe des ACG du

second ordre, dans lesquelles les termes du langage abstrait sont de la forme $q_1 \multimap \dots \multimap q_k \multimap p$ o  les q_i sont des types atomiques, ce probl me de l'appartenance est d cidable en temps polynomial.

2.2 Extension par les produits d pendants

L'architecture des ACG ne d pend pas de la logique sur laquelle elles reposent. Contrairement aux autres constructions propos es par Philippe de Groote et Sarah Maarek (de Groote & Maarek, 2007)(implication non lin aire, produit cart sien, union disjointe et type unit) pour  tendre le syst me de type, le produit d pendant ne correspond pas   une primitive de la logique lin aire. Il vient de la th orie intuitionniste des types de Martin-L f (Martin-L f, 1973) et a  t  utilis  dans le contexte de la logique lin aire par Cervesato et Pfenning (Cervesato & Pfenning, 2002). Le produit d pendant permet d'introduire des types qui d pendent de termes. Le formalisme ainsi  tendu est Turing-complet, et le probl me de l'appartenance est donc ind cidable (de Groote *et al.*, 2007). Il reste    tudier quels fragments utilisant le pouvoir d'expression ainsi gagn  restent d cibles. Cet exemple montre comment utiliser les types d pendants pour instancier un groupe nominal par son genre.

gender = $\{m, f\} : type$ (vocabulaire abstrait)
NP, N : (**gender**) *type*
S : *type*

$village_c : \mathbf{N} m$
 $ville_c : \mathbf{N} f$
 $un_c : (\Pi x : gender) \mathbf{N} x \multimap \mathbf{NP} x$
 $joyeux_c : (\Pi x : gender) \mathbf{N} x \multimap \mathbf{N} x$
 $dormir_c : (\Pi x : gender) \mathbf{NP} x \multimap \mathbf{S}$

gender = $\{m, f\} : type$ (vocabulaire objet)
string : *type*
ville joyeux, joyeuse, un, une, ... : **string**

$gender \longrightarrow gender$
 $\mathbf{N}, \mathbf{NP} \longrightarrow \lambda x. \mathbf{string}$ (lexique)
 $\mathbf{S} \longrightarrow \mathbf{string}$
 $ville_c \longrightarrow ville$
 $village_c \longrightarrow village$
 $un_c \longrightarrow \lambda g. \lambda x. \{m \rightarrow un, f \rightarrow une\} g + x$
 $joyeux_c \longrightarrow \lambda g. \lambda x. x + \{m \rightarrow joyeux, f \rightarrow joyeuse\} g$
 $dormir_c \longrightarrow \lambda g. \lambda x. x + dort$

On peut v rifier que le terme suivant appartient au langage abstrait.

$$\underbrace{\text{dormir}_c f \left(\underbrace{\text{un}_c f \left(\overbrace{\text{joyeux}_c f \text{ville}_c}^{\text{NP } f} \right)}_{\text{N } f} \right)}_S$$

On en déduit que son image par le lexique appartient au langage objet

$$\text{dormir}_c f (\text{un}_c f (\text{joyeux}_c f \text{ville}_c))$$

$$\longrightarrow \text{une ville joyeuse dort}$$

3 Mouvements grammaticaux explicites

Les mouvements grammaticaux explicites sont ceux visibles au niveau de surface du langage : ce sont les extractions de constituants syntaxiques par des constructions comme les interrogatives en *qu*, les relatives, ou les semi-clivées en français. On s'intéressera ici à la modélisation des contraintes d'extraction par les constructions relatives dans la langues française, ces contraintes étant très proches pour les autres mouvements explicites (Abeillé, 2002).

3.1 Propositions relatives en français dans les ACG

Dans les ACG le pronom relatif sujet *qui* serait représenté au niveau abstrait par le terme

$$\text{qui}_c : (\text{NP} \multimap \text{S}) \multimap \text{N} \multimap \text{N}$$

Le premier argument correspond à la suite de la proposition relative : c'est une phrase à laquelle il manque un sujet. Le second argument correspond au nom complété par cette proposition. L'image de cette constante par le lexique serait alors : $\lambda S. \lambda n. n + \text{qui} + (S \in)$. On peut par exemple vérifier que la phrase suivante appartient au langage objet de surface :

$$\text{dormir}_c (le_c (\text{qui}_c (\lambda x. \text{aimer}_c \text{Marie}_c x) \text{homme}_c))$$

$$\longrightarrow \text{l'homme qui aime Marie dort}$$

Il faut noter que le système de type, contrairement à la tradition des grammaires catégorielles, n'opère aucun contrôle sur l'ordre des mots dans les ACG. Cet ordre des mots est géré par le lexique. On n'a donc pas ici de problème à générer des extractions non-périphériques (*l'homme que Marie aime passionnement dort*) ou à distance arbitraire (*l'homme que Jean dit que Pierre croit que Marie aime dort*). D'un autre côté, sans contrôle sur la construction de nos termes abstraits, on rencontre plusieurs phénomènes de surgénération. Le premier est lié à l'absence de contraintes sur le rôle du constituant grammatical extrait. Rien n'empêche en effet de reconnaître *l'homme qui Marie aime dort* de manière similaire à l'exemple initial. L'autre contrainte

qui doit  tre prise en compte est l’existence d’ lots d’extraction   l’int rieur desquels l’extraction est impossible depuis l’ext rieur. Les propositions relatives elle-m mes sont un exemple d’ lot d’extraction, et en effet *L’homme que ma soeur qui aime dort* ne devrait pas  tre reconnu comme une phrase de la langue fran aise. Pourtant le terme

*le (que ($\lambda x.dormir$ (ma (qui ($\lambda y.aime$ x y) *soeur*))) *homme*)*

fait bien partie de notre langage abstrait. Les propositions compl tives ne sont, elles, des  lots que pour l’extraction de sujet : *l’homme qui Marie crois que vient* est grammaticalement incorrect.

3.2 Le probl me dans quelques autres formalismes

Les ph nom nes d’extraction ont  t  consid r s sous de nombreux angles diff rents. Examinons rapidement les solutions adopt es dans quelques formalismes voisins aux ACG.

Les Tree Adjoining Grammars permettent de bien rendre compte du ph nom ne (Abeill , 2002). Le sch ma de formation des arbres  l mentaires   compl ment relativis  modifie les arbres   racine phrastique pour placer le compl ment en t te de phrase avec un trait <qu>. Ce compl ment pouvait se trouver   profondeur arbitraire dans l’arbre initial puisqu’il apparait sur le sch ma sous un lien de dominance sous-sp cifi . Dans le cas d’une proposition relative, l’arbre obtenu est domin  par un noeud de cat gorie N qui domine un autre noeud de cat gorie N d crivant l’ant c dent. Dans tous les cas un noeud interm diaire phrastique est ajout  : il permet avec l’op rateur d’adjonction d’ajouter un nombre arbitraire de constructions comme *Marie croit que*. L’extraction du sujet est trait e de mani re similaire mais distincte.

Dans les grammaires d’interaction, chaque pronom relatif est d crit par un arbre dans lequel le compl ment extrait apparait mais est morphologiquement vide. Cette trace du compl ment pourra donc  tre unifi e avec les autres arbres o  le compl ment est d crit, mais elle n’appara tra pas en surface. Afin de permettre une d pendance   distance arbitraire entre le pronom relatif et le compl ment extrait, la trace est surmont e d’une relation de d pendance sous-sp cifi e. Cependant, pour bloquer la travers e d’ lots d’extraction, cette relation est contr l e : elle ne peut franchir que des noeuds  tiquet s S et de fonction objet ou nulle (contraintes satisfaites par *que Marie aime* par exemple). L’arbre d crivant le pronom relatif sujet se distingue par l’absence de trace : le pronom assure la fonction de sujet. (Perrier, 2007).

Les grammaires cat gorielles s’appuient traditionnellement sur le calcul de Lambek. Le connecteur implicatif est double : le / prend son argument directement   sa droite et le \ directement   sa gauche. L’ordre des mots est donc g r  dans le syst me de type. Ce syst me est trop rigide pour les constructions relatives puisqu’il emp che les extractions non-p riph riques : l’argument d’un pronom relatif est une phrase   laquelle il manque un constituant   sa droite ou   sa gauche seulement. Une possibilit  est d’ajouter un constructeur \uparrow et sa r gle d’introduction, $A\uparrow B$  tant le type d’un terme qui compl t  quelque part d’un terme de type B serait de type A. On peut alors donner au pronom *qui* le type $N\backslash N/(S\uparrow NP)$. On retrouve   ce moment la m me

surgénération que dans les ACG. La modélisation des îlots d'extraction peut se faire à l'aide de constructeurs modaux rattachés à l'extracteur et à la trace qui doivent se combiner tandis que les îlots d'extraction jouent le rôle de portes verrouillées à ces constructeurs (Moortgat, 1996), (Carpenter, 1997). C'est également l'approche présente dans les Lambda Grammars (Muskens, 2007). On verra que c'est de cette technique qu'on s'approchera le plus.

Dans les Convergent Grammars, les pronoms relatifs (et les autres constituants opérant des mouvements explicites) sont affectés d'une catégorie spéciale $G[A,B,C]$ indiquant que ce terme se combine avec un terme de catégorie A contenant une variable libre de type B et renvoie un terme de type C tout en liant cette trace. Cette catégorie spéciale permet de ne construire aucun terme par lambda-abstraction. Il suffit donc de vérifier que des variables libres ne sortent pas des îlots d'extraction.

4 Modélisation des contraintes d'extraction dans les ACG étendues

Le système de type étendu va nous permettre de contrôler la construction des termes du langage abstrait dans le respect des contraintes de la langue. Le premier terme dont doivent dépendre les types de nos constantes est un marqueur de cas, qui vérifiera le rôle des constituants extraits (comme on ne s'intéresse ici qu'à la signature abstraite on omettra d'indicer chaque nom de constantes par c).

$\mathbf{cas} = \{nom, acc, gen, comp\} : \text{type}$
 $\mathbf{NP}, \mathbf{N} : (cas) \text{ type}$
 $\mathbf{S} : \text{type}$
 Jean, Marie, Paul, Laurent : $(\Pi c : \mathbf{cas}) \mathbf{NP} \ c$
 chien, maison, homme : $(\Pi c : \mathbf{cas}) \mathbf{N} \ c$
 un, le : $(\Pi c : \mathbf{cas}) \mathbf{N} \ c \multimap \mathbf{NP} \ c$
 aimer, rencontrer : $\mathbf{NP} \ acc \multimap \mathbf{NP} \ nom \multimap \mathbf{S}$
 dormir : $\mathbf{NP} \ nom \multimap \mathbf{S}$
 qui : $(\Pi c : \mathbf{cas}) (\mathbf{NP} \ nom \multimap \mathbf{S}) \multimap \mathbf{NP} \ c \multimap \mathbf{NP} \ c$

La phrase suivante n'appartient plus au langage : *aimer* contraint la variable x à prendre comme argument le cas *acc*, et *qui* le cas *nom*. Le terme ne peut donc pas être typé.

$dormir_c (le_c \ nom \ (qui_c \ (\lambda x. aimer_c \ x \ acc \ Marie_c \ nom) \ homme_c \ nom))$

$\longrightarrow l'homme \ qui \ Marie \ aime \ dort$

On introduit un autre marqueur, de type **ext** pour extraction. L'idée est que les pronoms relatifs forcent les propositions relatives à étiqueter la variable trace de l'extraction par ce marqueur à 0. Ce marqueur est propagé aux niveaux syntaxiques supérieurs mais ne peut pas franchir les îlots d'extraction. Il reste alors au pronom relatif de contrôler que la proposition relative est bien étiquetée par 0. Une troisième valeur du marqueur permet d'indiquer qu'un îlot à l'extraction

de sujet a  t  franchit. Cette valeur interm diaire sera donc  galement accept e par tous les pronoms sauf le *qui* extracteur de sujet.

ext = {0, 1, 2} : type

NP : (ext) (cas) type

N : (ext) (cas) type

S : (ext) type

S* : (ext) type

chien, maison, homme : ($\Pi c : \mathbf{cas}$) **N** 2 *c*

un, le, mon : ($\Pi c : \mathbf{cas}$) **N** 2 *c* \multimap **NP** 2 *c*

marcher, dormir : ($\Pi y : \mathbf{ext}$) **NP** *y nom* \multimap **S** *y*

aimer, rencontrer : ($\Pi y_1, y_2 : \mathbf{ext}$) **NP** *y_1 acc* \multimap **NP** *y_2 nom* \multimap **S** *min (y_1 y_2)*

qui : ($\Pi c : \mathbf{cas}$) (**NP** 0 *nom* \multimap **S** 0) \multimap **N** 2 *c* \multimap **N** 2 *c*

que : ($\Pi y : \mathbf{ext}, c : \mathbf{cas}$) (**NP** 0 *acc* \multimap **S** *y - 1*) \multimap **N** 2 *c* \multimap **N** 2 *c*

que' : ($\Pi y : \mathbf{ext}$) **S** *y* \multimap **S** * *max(y, 1)*

(je crois *que* l'id e est bonne)

croire, dire : ($\Pi y_1, y_2 : \mathbf{ext}$) **S** * *y_1* \multimap **NP** *y_2 nom* \multimap **S** *min (y_1 y_2)*

On peut v rifier que ce terme n'est plus typ  par la grammaire : (on omet d' crire les termes du produit d pendant pour une meilleure lecture)

L'homme que ma soeur qui aime dort

*le (que ($\lambda x.$ dormir (*ma* (*qui* ($\lambda y.$ aime *x y*) soeur)))) homme)*

En effet les deux variables qui jouent le r le de traces *x* et *y* doivent porter le marqueur 0 (exigence du type des pronoms relatifs). Ce marqueur va  tre consomm  par le *qui*, et le marqueur d'extraction qui arrivera au *que* ne pourrait  tre que 2, ce qui rend le terme intypable. A l'inverse le terme suivante reste toujours typable :

L'homme que ma soeur qui dort aime

$$\begin{array}{c} \text{S } 0 \\ \hline \text{le } (\text{que } (\lambda x. \text{aime } \underbrace{x}_{\text{NP } 0 \text{ acc}} (\text{ma } (\text{qui } (\lambda y. \text{dort } \underbrace{y}_{\text{NP } 0 \text{ nom}}) \text{soeur})))) \text{homme})) \\ \hline \text{N } 2 \text{ c} \end{array}$$

Enfin, on v rifie que ce groupe nominal n'est plus typable :

Pierre qui je crois que vient

$$\begin{array}{c} \text{S } * 1 \\ \hline \text{qui}_2 (\lambda x. \text{croire } (\text{que'} (\text{venir } x))) \text{je}) \text{Pierre} \\ \hline \text{NP } 0 \text{ nom} \multimap \text{S } 1 \end{array}$$

Le marqueur 0 de la variable trace devient 1 en passant par le *je crois que*. Le terme r sultant n'a donc pas le bon type pour  tre argument de *qui*.

Pierre que je crois que Marie aime

a la même forme de dérivation de type, à la différence de *que* qui, lui, accepte ce type pour son argument.

Ce contrôle s'étend aisément aux autres pronoms relatifs et aux autres îlots d'extraction, comme les *si* ou les propositions infinitives. Les îlots lexicalisés, comme le verbe *hurler* ne posent pas plus de problèmes. De plus les autres phénomènes de mouvements explicites (interrogatives en *qu*, semi-clivées) peuvent utiliser le même marqueur et la même structure de contrôle. Notons enfin que l'on pourrait avoir démultiplié les entrées lexicales plutôt qu'utiliser les types dépendants, on est donc resté dans un fragment réversible des ACG.

5 Conclusion

Un atout des ACG est de proposer directement un contrôle sur les structures de dérivations, qui constituent notre langage abstrait. Il est ainsi possible de modéliser des phénomènes linguistiques sans quitter le formalisme ni le lambda-calcul typé. Cette technique de modélisation des contraintes à l'extraction propose une illustration pratique des possibilités offertes par l'extension du système de type des ACG.

D'autres phénomènes de déplacement peuvent être modélisés selon le même schéma. En particulier, les mouvements grammaticaux implicites, qui sont visibles au niveau de la réalisation sémantique, présentent des caractéristiques proches des mouvements présentés dans cet article, sur la nature des extractions (opérées dans ce cas par les prises de portée des quantificateurs) comme sur celle des îlots d'extraction (Ruys & Winter, 2008).

Références

- ABEILLÉ A. (2002). *Une grammaire électronique du français*. CNRS Editions.
- CARPENTER B. (1997). *Type-Logical Semantics*. The MIT Press.
- CERVESATO I. & PFENNING F. (2002). A linear logical framework. *Information and Computation*, **179**(1), 19–75.
- DE GROOTE P. (2001). Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, p. 148–155.
- DE GROOTE P. & MAAREK S. (2007). Type-theoretic extensions of abstract categorial grammars. In *New Directions in Type-Theoretic Grammars, proceedings of the workshop*, p. 18–30.
- DE GROOTE P., YOSHINAKA R. & MAAREK S. (2007). On two extensions of abstract categorial grammars. In N. DERSHOWITZ & A. VORONKOV, Eds., *Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15-19, 2007, Proceedings*, volume 4790 of *Lecture Notes in Computer Science*, p. 273–287 : Springer.

- MARTIN-LÖF P. (1973). An intuitionistic theory of types : Predicative part. In F.ROSE & J.SHEPERDSON, Eds., *Logic Colloquium '73*, p. 73–118.
- MOORTGAT M. (1996). Categorical type logics. In J. VAN BENTHEM & A. TER MEULEN, Eds., *Handbook of Logic and Language*, p. 93–177. Amsterdam : Elsevier Science Publishers.
- MUSKENS R. (2007). Separating syntax and combinatorics in categorial grammar. *Research on language and computation*.
- PERRIER G. (2007). A french interaction grammar. In *6th International Conference on Recent Advances in Natural Language Processing, Borovets, Bulgaria, 2007.*, p. 463–467.
- POLLARD C. (2007). The logic of pied-piping. Presentation at the Colloquium in Honour of Alain Lecomte.
- POLLARD C. (2008). An introduction to convergent grammar. Presentation at Calligramme Seminar.
- RUYS E. & WINTER Y. (2008). Quantifier scope in formal linguistics. To appear in D. Gabbay (ed.), *Handbook of Philosophical Logic - Second Edition*.
- SALVATI S. (2005). *Problèmes de filtrage et problèmes d'analyse pour les grammaires catégorielles abstraites*. PhD thesis, Institut National Polytechnique de Lorraine.